```
-- AUTOR: Javier Campos. Modificado por Elvira Mayordomo Cámara
-- PROYECTO: módulo de implementación del TAD listas con acceso por posición
-- FICHERO: listasap.adb
-- FECHA: 31-X-96; 16-10-03; 22-10-03; 11-11-03

with unchecked_deallocation;
package body listasap is
      procedure free is new unchecked_deallocation(unDato,ptUnDato);

      procedure creaVacia(l:out lista) is
      -- Post: l=listaVacía
      -- coste en tiempo O(1)
      begin
            l:=(null, null,0);
      end creaVacia;

      procedure aniadirIzq(e:in elemento; l:in out lista) is
      -- Pre: l=l0
      -- Post: l=+izq(e,l0)
      -- coste en tiempo O(1)
```

```
begin
        l.prim:=new unDato'(e,l.prim);
        if l.ult=null then
                l.ult:=l.prim;
        end if;
        l.n:=l.n+1;
end aniadirIzq;

procedure concatenar(l1:in out lista; l2:in lista) is
-- Pre: l1=l0
-- Post: l1=l0 & l2
-- coste en tiempo O(1)
-- Atención: No se realiza copia, la lista l2 ya no se puede usar ya que forma parte
-- de l2
begin
        if esVacia(l1) then
                l1.prim:=l2.prim;
        else
                l1.ult.sig:=l2.prim;
        end if;
```

```
        l1.ult:=l2.ult;
        l1.n:=l1.n+l2.n;
end concatenar;

function longitud(l:lista) return natural is
-- Post: longitud(l)= longitud(l)
-- coste en tiempo O(1)
begin
        return l.n;
end longitud;

function esta(e:elemento; l:lista) return boolean is
-- Post: esta(l)= está(l)
-- coste en tiempo: para una lista de n datos tarda O(n)
aux:ptUnDato;
begin
        aux:=l.prim;
        while aux/=null and then aux.dato/=e loop
```

```
            aux:=aux.sig;
        end loop;
        return (aux/=null);
end esta;

function esVacia(l:lista) return boolean is
-- Post: esVacia(l)= esvacía(l)
-- coste en tiempo O(1)
begin
        return l.n=0;
end esVacia;

procedure creaUnitaria(e:in elemento; l:out lista) is
aux:lista;
-- Post: l=+izq(e,listaVacía)
-- coste en tiempo O(1)
begin
        aux.prim:=new unDato'(e,null);
        aux.ult:=aux.prim;
        aux.n:=1;
```

```
        l:=aux;
end creaUnitaria;

procedure aniadirDch(l:in out lista; e:in elemento) is
-- Pre: l=l0
-- Post: l=+dcha(l0, e)
-- coste en tiempo O(1)
begin
        if esVacia(l) then
                creaUnitaria(e,l);
        else
                l.ult.sig:=new unDato'(e,null);
                l.ult:=l.ult.sig;
                l.n:=l.n+1;
        end if;
end aniadirDch;

procedure insertar(l:in out lista; i:in natural; e:in elemento) is
-- Pre: l=l0
-- Post: l=insertar(l0,i,e)
```

```
-- coste en tiempo O(i)
aux:ptUnDato; indSig:natural;
begin
      if i=1 then
            aniadirIzq(e,l);
      elsif i=longitud(l)+1 then
            aniadirDch(l,e);
      elsif (1<i) and (i<longitud(l)+1) then
            aux:=l.prim;
            indSig:=2; -- la posición de insertar después de aux
            while indSig<i loop
                  aux:=aux.sig;
                  indSig:=indSig+1;
            end loop;
            aux.sig:=new unDato'(e,aux.sig);
            l.n:=l.n+1;
      end if;
end insertar;

procedure eliminarIzq(l:in out lista) is
```

```
-- Pre: l=l0
-- Post: l=borrar(l0,1)
-- coste en tiempo O(1)
aux:ptUnDato;
begin
      if not esVacia(l) then
            aux:=l.prim;
            l.prim:=l.prim.sig;
            if l.prim=null then  -- he borrado el único elemento
                  l.ult:=null;
            end if;
            l.n:=l.n-1;
            free(aux);
      end if;
end eliminarIzq;

procedure eliminarDch(l:in out lista) is
aux:ptUnDato;
-- Pre: l=l0
-- Post: l=borrar(l0,longitud(l0))
```

```
-- coste en tiempo: para una lista de n datos tarda O(n)
begin
      if not esVacia(l) then
            if l.prim=l.ult then
                  free(l.prim);
                  creaVacia(l);
            else
                  aux:=l.prim;
                  while aux.sig/=l.ult loop
                        aux:=aux.sig;
                  end loop;
                  aux.sig:=null;
                  free(l.ult);
                  l.ult:=aux;
                  l.n:=l.n-1;
            end if;
      end if;
end eliminarDch;

procedure borrar(l:in out lista; i:in natural) is
```

```
-- Pre: l=l0
-- Post: l=borrar(l0,i)
-- coste en tiempo O(i)
aux,sigAux:ptUnDato; indSig:natural;
begin
      if i=1 then
            eliminarIzq(l);
      elsif i=longitud(l) then
            eliminarDch(l);
      elsif (1<i) and (i<longitud(l)) then
            aux:=l.prim;
            indSig:=2;
            while indSig<i loop
                  aux:=aux.sig;
                  indSig:=indSig+1;
            end loop;
            sigAux:=aux.sig;
            aux.sig:=sigAux.sig;
            free(sigAux);
            l.n:=l.n-1;
```

```
        end if;
end borrar;

function observar(l:lista; i:positive) return elemento is
-- Pre: (1<=i)∧(i<=longitud(l))
-- Post: observar(l,i)=l[i]
-- coste en tiempo O(i)
aux:ptUnDato; ind:positive;
begin
        aux:=l.prim;
        ind:=1;
        while ind<i loop
                aux:=aux.sig;
                ind:=ind+1;
        end loop;
        return aux.dato;
end observar;

function posicion(e:elemento; l:lista) return natural is
-- Post: posicion(e,l)=posición(e,l)
```

```
    -- coste en tiempo: para una lista de n datos tarda O(n)
    aux:ptUnDato; i:positive;
    begin
        aux:=l.prim;
        i:=1;
        while aux/=null and then aux.dato/=e loop
            aux:=aux.sig;
            i:=i+1;
        end loop;
        if aux/=null then
            return i;
        else
            return 0;
        end if;
    end posicion;

end listasap;
```